Zen in the Art of Cost Estimation

Ricardo Valerdi
Massachusetts Institute of Technology
rvalerdi@mit.edu

Abstract. Engineering cannot wait until all phenomena are explained. Engineers may work effectively, often for centuries, with heuristics. This paper provides over thirty heuristics that have been inspired by the development and application of a systems engineering cost estimation model. The objective of this paper is to present such heuristics in a simple manner so that they can benefit those that develop, calibrate, and use cost models.

Introduction

This paper provides a collection of heuristics, or small nuggets of wisdom, to help systems engineers along the road of cost estimation in the same way Zen Buddhism encourages seeing deeply into the nature of things by direct experience. The Zen monks taught that in every menial task there is something to be learned and something that the student can later meditate on. The Zen monks believe that one can gather knowledge from all things in life. While Zen began in China in the 6th century and spread to Japan, Vietnam, Korea, and eventually the Western world, several authors have made the connection between Zen and other activities such as motorcycle maintenance (Pirsig 1974), flying (Shade 1975), golf (Hebron 1990), writing (Bradbury 1992), fatherhood (Lewis 1997), comedy (Sankey 1998), flower arrangement (Herrigel 1999), and even tennis (Koran 2002). Arguably the most popular example is the text *Zen in the Art of Archery* (Herrigel 1948) which was written by German philosopher Eugen Herrigel (1884–1955). Since its first German edition in 1948, the book has been translated into several foreign languages (English, Japanese, Portuguese, etc.) and has been continually reprinted as one of the best-selling works on Japanese culture. As a result, the term “Japanese archery” evokes thoughts of spiritual training and Zen spirituality (Yamada 2001).

Heuristics

A heuristic is a method to help solve a problem, commonly informal. It is particularly used for a method that often rapidly leads to a solution that is usually reasonably close to the best possible answer. Heuristics are "rules of thumb", educated guesses, intuitive judgments or simply common sense. In more precise terms, heuristics stand for strategies using readily accessible, though loosely applicable, information to control problem-solving in human beings and machines. Heuristics are common in psychology, philosophy, law, and engineering. A commonly used heuristic is:

If you are having difficulty understanding a problem, try drawing a picture.

In psychology, heuristics are simple, efficient rules, hard-coded by evolutionary processes. Heuristics have been proposed to explain how people make decisions, come to judgments, and solve problems, typically when facing complex problems or incomplete information. These rules work well under most circumstances, but in certain cases lead to systematic cognitive biases. Because heuristics are fallible, it is important to understand their limitations. They are only intended to be used as aids in order to make quick estimates and preliminary process designs. Their applicability is also limited by the context under which they were derived.

Heuristics in Systems Architecting & Engineering

In engineering, a heuristic is an experience-based method that can be used as an aid to solve specific problems (Endres & Rombach 2003). By using heuristics, time can be reduced when solving problems, which may be very valuable. While heuristics are not always scientifically validated, they represent many years of empirical observations of experienced professionals packaged in the form of concise wisdom.

One area of engineering that has benefitted from heuristics is the field of systems architecting as a result of Eb Rechtin's (1926–2006) book *Systems Architecting: Creating and Building Complex Systems* (Rechtin 1991). The book contains a spectrum of more than 100 heuristics that provide insights in the areas of systems architecting, development, testing, and organizations which were derived from Rechtin's 40 year career at NASA/Jet Propulsion Laboratory, Defense Advanced Research Projects Agency, the Department of Defense, the Aerospace Corporation, and Hewlett-Packard. An example heuristic provided by Rechtin is:

> *Don't assume that the original statement of the problem is necessarily the best, or even the right one.*

This heuristic suggests the need for deeper analysis of solutions. It works well when problem and solution are not too far apart, that is, when a few leaps of imagination may bring problem and solution together.

Heuristics-Based Cost Model

COSYSMO (the Constructive Systems Engineering Cost Model) is a parametric model that was developed for the purpose of estimating the cost of systems engineering effort in large complex systems. The model enables the estimation of systems engineering effort through an assessment of system characteristics such as:

- Number of System Requirements
- Number of System Interfaces
- Number of System-Specific Algorithms
- Number of Operational Scenarios

Additionally, the systems engineering aspect of projects can be assessed for the following cost/performance characteristics:

- Requirements Understanding
- Architecture Understanding
- Level of Service Requirements
- Migration Complexity
- Technology Risk
- Documentation
- # and diversity of installations/platforms
- # of recursive levels in the design
- Stakeholder team cohesion
- Personnel/team capability
- Personnel experience/continuity
- Process capability
- Multi-site coordination
- Tool Support

These characteristics are representative of the most influential systems engineering cost drivers as defined by industry experts and validated by through historical data (Valerdi 2005).

COSYSMO was inspired by a number of Rechtin's heuristics which eventually led to a software implementation of the model – hereafter referred to as "the tool" – that was used by several aerospace and defense organizations in the United States. The tool is an important milestone because it enables the use of the model in a way that can be modified by different organizations to fit their needs. In the process of helping organizations tailor the model, a number of observations and lessons learned were captured which were eventually formalized into heuristics. These heuristics were tested across different organizations for their validity and were reworded for clarity and scope. User experience confirmed Rechtin's heuristics in many ways as will be discussed in the next section. The closed-loop relationships of these ideas are illustrated in Figure 1.

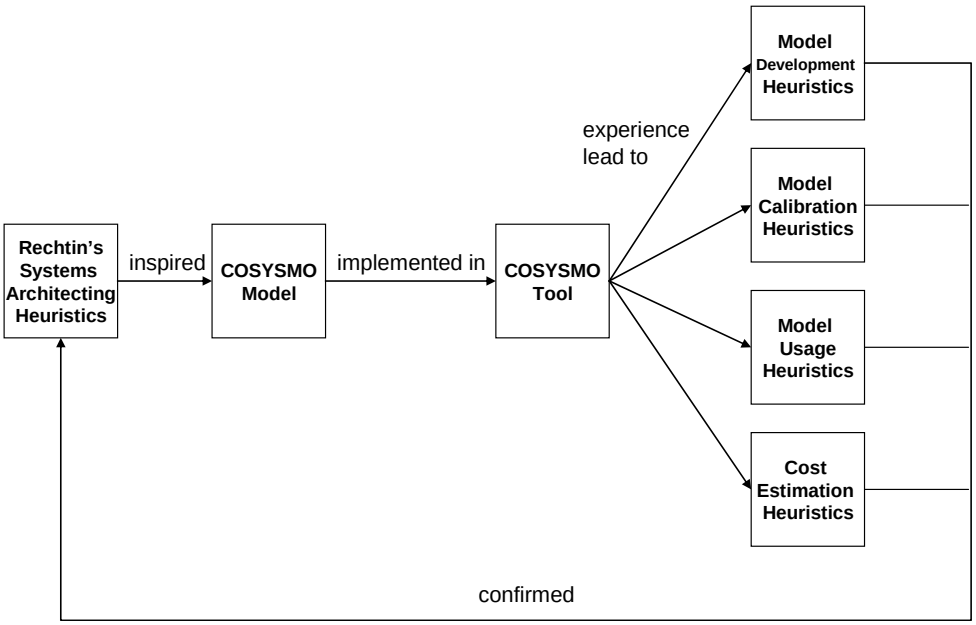

Figure 1. Experiential closed loop.

Rechtin's System Architecting Heuristics

The definitive text for all systems architecting and engineering students has been Rechtin's book *Systems Architecting: Creating and Building Complex Systems* (Rechtin 1991). Rechtin provides three elements that all systems architects need to be effective:

1. Know the engineering fundamentals on which each architecture is based. Common sense, derived from specific engineering fundamentals and the experience of other architects, is needed to reduce the search to practical dimensions.
2. Experience and judgment are necessary. Hands-on system problem solving is mandatory.
3. Acquire the insights gained from experience in the design laboratories on the job.

These elements, and their associated heuristics, were critical during the COSYSMO development process. Not only because they provided a starting point for the most important cost drivers in systems engineering, but also because they were a representation of decades of experience in complex systems. These were not only Rechtin's experience; but a collection of recorded lessons provided by is colleagues throughout his career.

Some of the heuristics that inspired specific parameters in COSYSMO are provided in Table 1.

Table 1. Relationship between Rechtin's heuristics and COSYSMO

Systems Architecting Heuristic (Rechtin 1991)	COSYSMO Driver
Success is defined by the beholder (understand his or her requirements) not by the architect	Number of System Requirements
The greatest leverage in systems architecting is at the interfaces	Number of Major Interfaces
One person's architecture is another person's system is another person's component	Architecture Understanding
There will always be friction on the "illity" boundary	Level of Service Requirements
If you don't understand the existing system you can't be sure you are re-architecting a better one	Migration complexity
"Proven" and "State of the Art" are mutually exclusive qualities	Technology Maturity
Amid a wash of paper, a small number of documents become critical pivots around which every project manager revolves	Documentation to Match Life Cycle Needs
Organize personnel tasks to minimize the time individuals spend interfacing	# and Diversity of Installations & Platforms
A strong coherent constituency is essential	Stakeholder Team Cohesion

The relationship between Rechtin's heuristics and a selected group of the COSYSMO drivers illustrates the important evolution from a rule of thumb to a parameter that needs to be operationalized. The process of developing a rating scale and describing each parameter in ways that it can be quantified and measured is a natural step that takes place as disciplines mature. The next section provides a detailed description of the heuristics that resulted from the application of COSYSMO to multiple contexts.

Four Categories of Heuristics

As shown in Figure 1, the transition of ideas from heuristics -> model -> tool -> heuristics was a process fueled by usage experience and feedback. This iterative process led to the creation and refinement of over thirty heuristics. In order to better appreciate their applicability, the heuristics are organized into four categories: model development, model calibration, model usage, and estimation.

The methodology for creating heuristics followed four general guidelines. First, there had to be agreement among experts that the heuristic was useful and correct. Second, the heuristic had to stand the test of time, that is, show its applicability over time. Third, the heuristic had to be resilient across different scenarios beyond the one under which it was created. Fourth, the heuristic had to demonstrate its value by reoccurring more than once and not be considered as

obvious by everybody, particularly people who are new to the field. Situations that were not likely to be repeated did not warrant their own heuristic. Some heuristics have been reworded from their original form in order to serve a broader audience or clarify their intent. This has been done with the care of maintaining the original intention of the heuristic.

Model Development Heuristics

The process of developing a cost model provides numerous insights into how certain phenomena can be modeled. Some of these heuristics are generalizable to modeling in general while others are specific to system engineering cost modeling. This section contains 5 heuristics.

Heuristic #1: More parameters increase the explanatory power of the model, but too many parameters make the model too complex to use and difficult to calibrate.

Albert Einstein once said that we should "Make everything as simple as possible, but not simpler." This philosophy carries over to the model development process which involves the selection of the parameters that (1) have the best explanatory power; and (2) are logically the most appropriate for representing systems engineering cost. This down selection is often made with the support of statistical tests such as p-value and t-test (Cook & Weisberg 1999) but discussions with users across a broad range of interests often leads to the addition of parameters in the model that are important to their particular situation. While the addition of parameters results in broader applicability of the model and often (but not always) can result in increased explanatory power, there are disadvantages from a usability standpoint. Models with more parameters are often more complex to use and require more data to be validated. It is necessary to determine the marginal benefit contributed by each parameter.

Heuristic #2: Break the problem and analysis into phases over time; the right amount of granularity is important.

In the same way engineering problem solving encourages the decomposition of problems to make them more manageable, cost modeling lends itself to this approach because of the complex task of capturing cost over the system life cycle. Cost modelers have found it beneficial to match the life cycle phases for which cost is reported to the life cycle phases defined in industry standards (ANSI/EIA. 1999), government policy (NASA 2002), or commercial practices (Royce 1998). There is also an important balance here in terms of the level of detail provided in a cost model. One natural limitation for the level of detail is the availability of data. This prevents the development of a cost model that has too much detail and does not match the level in which its users are functioning. Equally important is the need for enough detail in the cost model so that the user can understand how the cost estimate is decomposed across life cycle phases.

Heuristic #3: Let available data drive the application boundaries of the model.

The scope of cost models is often driven by the practitioner community that is willing to support it and the availability of data that can be provided to validate it. In the case of COSYSMO, the involvement of INCOSE and associated aerospace & defense organizations played an important role in determining the operational concept of the model. The most obvious artifacts of this influence are the terminology used in COSYSMO and the standards used to define the model. In addition, the data obtained from organizations collectively defined the region of operation for the model.

Heuristic #4: Design the rating scale according to the phenomenon being modeled.

This heuristic can be generalized to parametric model development activities because it

involves the process of developing rating scales for cost drivers. The impact of project characteristics on systems engineering can be represented by cost drivers with different polarities; positively defined parameters translate to a cost savings while negatively defined parameters translate to a cost penalty. Furthermore, cost drivers that are positively phrased – for example, *requirements understanding* – need to include a dynamic range that spans across three regions of impact: negative impact, no impact, and positive impact. Each region is defined with descriptive guidance that allows the user to determine which level is most appropriate for describing the system they wish to estimate. The ratings scale for the *requirements understanding* parameter is provided in Table 2.

Table 2. Rating scale for requirements understanding parameter

Very Low	Low	Nominal	High	Very High
Poor: emergent requirements or unprecedented system	Minimal: many undefined areas	Reasonable: some undefined areas	Strong: few undefined areas	Full understanding of requirements, familiar system

As demonstrated in the rating scale, having emergent requirements or an unprecedented system is characterized with a "Very Low" rating for this driver. The "Nominal" case represents no impact of this cost driver on systems engineering cost. The "High" and "Very High" rating levels describe situations where systems engineering cost would be reduced based on the familiarity with the system. A rating scale with an equal number of positive and negative statements (called balanced keying) reduces the presence of acquiescence bias, the agreement with statements as presented, since acquiescence of positively defined items will balance the acquiescence of negatively defined items.

Heuristic #5: Some system characteristics are more likely to be cost penalties than cost savings.

There are certain characteristics of systems that, when present, introduce a significant amount of work for systems engineers throughout the life cycle. In the situations, it is necessary to depart from a balanced view of a rating scale and instead use a single-sided representation of the cost driver. Rather than centering the rating scale at "Nominal" as in the previous example, a situation where there is no opportunity for cost savings requires a rating scale that describes the cost penalties associated with the existence of such a driver. The best example is the *migration complexity* cost driver. The corresponding rating scale is provided in Table 3.

Table 3. Rating scale for migration complexity parameter

	Nominal	High	Very High	Extra High
Legacy contractor	Self; legacy system is well documented. Original team largely available	Self; original development team not available; most documentation available	Different contractor; limited documentation	Original contractor out of business; no documentation available
Effect of legacy system on new system	Everything is new; legacy system is completely replaced or non-existent	Migration is restricted to integration only	Migration is related to integration and development	Migration is related to integration, development, architecture and design

An additional property that makes this rating scale unique is its representation of *migration complexity* through the use of two similar but independent viewpoints: legacy contractor and effect of legacy system on new system. User experience showed that this cost driver was more adequately described by these viewpoints for the types of projects being estimated in COSYSMO. Despite the fact that this rating scale is a slight deviation from the typical odd-numbered items used Likert scaling (Likert 1932), it is believed that the descriptions provided for each choice are detailed enough for a cost estimator to make a reliable assessment of the project. Moreover, some of the possible distortions that exist in 5-level rating scales are not likely to have an effect on this rating scale. In particular, the central tendency bias, which causes respondents to avoid using extreme response categories, is avoided by the fact that the existence of legacy systems is binary. The social desirability bias, which is the tendency to portray an organization in a more favorable light, is also unlikely to have an effect on this rating scale because rating levels are easily verifiable with the actual circumstances of a systems engineering project. In other words, there is virtually no subjectivity in the descriptions of the rating levels.

Model Calibration Heuristics

In addition to the model development heuristics, there is a special class of heuristics that are specific to the calibration of cost models. These are a result of users summarizing their experience with tailoring the model to their organizational processes and developing local calibrations based on their historical data. Some of these heuristics are generalizable to calibration and data collection in general while others are specific to COSYSMO. This section contains 6 heuristics.

Heuristic #6: All calibrations are local.

Inspired by Massachusetts Representative Tip O'Neill's expression about all politics being local (O'Neill & Hymel 1995), this heuristic emphasizes the need to develop local calibrations across different organizations and business units. This is the most effective way to reflect an organization's way of doing business and the productivity of the systems engineering organization. Local calibrations have shown to improve a model's estimation accuracy for most parametric models (Boehm et al 2000).

Heuristic #7: Calibrations fix chronic errors in over- or underestimation.

This is a variation on the previous heuristic with a focus on the effects of calibration on model accuracy. It highlights the fact that if something is done consistently wrong, it is not wrong. This may sound counter-intuitive, but a chronic overestimation or underestimation of systems engineering can be fixed with a local calibration.

Heuristic #8: Be skeptical of data that you did not collect.

In the same way parents teach their children not to trust strangers; the same rule applies to data collection. Since data are rarely documented adequately for uses other than the ones for which they were intended for, they pose serious validity risks if they were collected by someone else. There may be readily available systems engineering data and it would be tempting to use. But experience advises against it unless one can determine precisely what systems engineering activities are included and excluded in the data reported.

Heuristic #9: For every parameter in the model, 5 data points are required for the calibration.

There are practical reasons as well as mathematical reasons for this heuristic. The practical reason is that the collection of additional data points acts as a training mechanism for

organizations. They get accustomed to the process of collecting data in a reliable fashion and begin to understand the model scope and definitions. Once the training threshold is reached, the statistical rules come into effect. From a statistical analysis standpoint, in order to achieve statistical significance the number of samples needs to exceed the number of parameters being calibrated. The more degrees of freedom available to perform statistical tests, the better. In particular, a ratio of 5 samples per parameter stems from the fact that it is the lowest sample size that provides a confidence greater than 90% (exactly 93.8%) and is referred to as the Rule of Five (Hubbard 2007).

Heuristic #10: Don't do more analysis than the data is worth.

In the business of cost model calibration, data are very hard to come by because of its sensitive nature. Once data are provided in a usable form the next significant challenge is the suite of statistical tests and associated analysis that can be done with it. This is not a matter of selecting the appropriate tests but rather whether the data warrants the necessary level of analysis. This partly an issue of sample size – addressed in the previous heuristic – and an issue of significant digits in the results. It is not appropriate to provide results of descriptive or inferential statistics at a precision that exceeds the goodness of the data. Just because the math can be done does not mean it should be done. Others refer to this as "paralysis by analysis."

Heuristic #11: You need less data than you think, you have more data than you think.

This heuristic is borrowed from Dough Hubbard's book *How to Measure Anything* (Hubbard 2007) where he describes the common perception of organizations that collecting data is difficult. To overcome the fear and overwhelming hesitation associated with model calibration it is helpful to clarify that the less data once has, the more beneficial the little bit of data that one can get becomes. In the case of COSYSMO, organizations need to focus on collecting data from their most representative projects. In addition, not all size and cost drivers are necessary to perform a calibration. This can be done with one or two of them, rather than four.

Model Usage Heuristics

In addition to the model calibration heuristics, there are a number of important heuristics having to do with the use of parametric cost models. These are a result of users providing insights from their experience using the model to estimate systems engineering projects. Some of these heuristics are generalizable to all types of cost models while others are specific to COSYSMO. This section contains 15 heuristics. Where indicated, some of these are variations of what others have said in different contexts. In these cases, explanations are provided to illustrate the relevance of the heuristic to model usage.

Heuristic #12: A model is not reality.

Rechtin reminds us that mathematical models, so elegant and pristine, can be particularly deceptive. A model is an abstraction of what the participants think (and hope) the end system and its environment will look like. What actually results is almost always different. The reliance on the model as the "silver bullet" (Brooks 1995) that will solve all problems is a risky one.

Heuristic #13: All models are wrong, but some of them are useful.

This heuristic comes from the heading in a book by George Box (1979) entitled *Robustness in the Strategy of Scientific Model Building*. The message is that models have many limitations but at the same time provide useful information for making inferences about the phenomenon being modeled. Beginning with the skeptical view that all models are wrong may sound negative but it may a healthy dose of realism that is necessary for those who rely too much on

models for the correct answer.

Heuristic #14: Begin with the end in mind.

This heuristic is inspired by the bestselling self-help text by Franklin Covey (1989) entitled *The Seven Habits of Highly Effective People*. It reminds users of cost models to estimate the number of requirements in a system as they will be at the end. In other words, the initial description of the system may not be an adequate representation of the final system. As a result, a judgment needs to be made to determine how many requirements will be in the system at time of delivery. This quantity will be a better predictor of the amount of systems engineering effort needed to do the job.

Heuristic #15: Requirements are king.

In the discipline of systems engineering, requirements are an important vehicle for articulating the functions of a system. From the standpoint of COSYSMO, requirements are the most important size driver in the model. The Cost Estimating Relationship in COSYSMO converts the three other size inputs (interfaces, algorithms, and operational scenarios) as "equivalent requirements." Getting this quantity right is the most important step in using the COSYSMO model for estimating systems engineering cost.

Heuristic #16: Not all requirements are created equal.

Requirements have different levels of complexity depending on the difficulty associated with their implementation and testing. This distinction of variable complexities of requirements is accounted for in the COSYSMO Cost Estimating Relationship by providing three possible categories for requirements: easy, nominal, and difficult. This approach proved to be a more accurate representation of system complexity.

Heuristic #17: Reuse is not free.

The economic impact of reuse was also an important phenomenon to consider when evaluating systems engineering complexity. Contrary to the idea that reused requirements have no additional systems engineering cost, it was demonstrated that (1) different classes of reuse existed, and (2) their influence on systems engineering cost varied by class. The classes defined were: new, reused, modified, deleted, and managed (Wang et al 2008).

Heuristic #18: Operational Scenarios may come first, but requirements will ultimately describe the system.

Early in the system lifecycle the system requirements may not be defined at a detailed level. However, the operational scenarios may be evident from project documentation. This heuristic suggests that operational scenarios are logical metrics to represent the systems engineering effort in the absence of requirements but eventually, after the system requirements are defined, the system's functions will be better represented by requirements.

Heuristic #19: Don't double dip.

An associated risk presented by the ability to count system functionality through the use of two independent size drivers (requirements and operational scenarios) is the possibility of duplication. Users may find it easier to count some aspects of the system as operational scenarios or interfaces but care must be taken to avoid counting these identical characteristics by the number of requirements. It is recommended that the user determine which driver to use to prevent to possibility of overestimating systems engineering effort.

Heuristic #20: Find your sea level.

This heuristic is inspired by a framework presented in the book *Writing Effective Use Cases*

by Alistair Cockburn (2000). It identifies the need for finding the most appropriate system level in which COSYSMO can be used. In particular, the level of the system in which requirements should be counted must be identified so that the systems engineering effort can be adequately estimated.

Heuristic #21: Nominal is the norm.

All cost drivers in COSYSMO have a default setting of "Nominal." In the process of rating the cost drivers, it is customary to leave the rating at its default level unless there is overwhelming evidence that it should be otherwise. Furthermore, in cases where a particular cost driver is at an intermediate level, say between "High" and "Very High", the preference should be to round towards "Nominal" (and select "High" in this case).

Heuristic #22: If you're estimating a large project, personnel capability is Nominal.

When a large project is being estimated it is safe to assume that a large staff of systems engineers will be assigned to the project. This results in a range of skill sets which, on average, operate at Nominal capabilities. Similar to the Law of Large Numbers, there is a tendency for the overall capability to be normally distributed and the different productivity and experience levels to balance each other. This should anchor the rating for *personnel capability* at Nominal.

Heuristic #23: Most of your off-Nominal cost drivers should match your last project.

Certain patterns begin to emerge after repeated use of the COSYSMO model. Organizations have noticed certain common settings that describe their usual customers, projects, tools, process, environment, etc. It is a good rule of thumb to compare any off-Nominal settings on cost drivers to previously estimated projects. Some commonly recurring patterns include *process capability* (typically set to "High") and *personnel continuity.* These are attributes of an organization that typically do not change over time.

Heuristic #24: If you're going to sin, sin consistently.

The scope of systems engineering in an organization is determined by what activities are included or excluded relative to the standard COSYSMO definitions. Once this has been defined, organizations must maintain the same approach for calibration and estimation. The approach may be unique but the key here is consistency. Organizations are encouraged to estimate with the same detail and scope of how they calibrate.

Heuristic #25: Use a combination of models to estimate total system cost.

The scope of some systems engineering activities spans beyond what is defined in COSYSMO. In this situation it is suggested that other models such as COCOMO II be incorporated to account for any software engineering activities that need to be included in the estimate. Similarly, there are hardware engineering models like PRICE-H and SEER-H that provide a useful complement to COSYSMO.

Heuristic #26: Avoid overlap between models.

The use of multiple models introduces the risk of overlap between them. Special attention needs to be taken to understand the amount of systems engineering effort already assumed in other models when combining them with COSYSMO.

Heuristic #27: Estimate using multiple methods (analogy, parametric, etc.)

Another useful best practice that is encouraged in industry guidebooks (NASA 2002) is the use of multiple estimation methods. This includes estimation by analogy, parametric models, expert assessment, and bottom up. This technique allows for triangulation of results that can improve the confidence in the estimate provided.

Estimation Heuristics

The final set of heuristics addresses the estimation process itself. It is important because of the heavy dependence on human judgment in the estimation process. This section has 4 heuristics.

Heuristic #28: Estimate early and often.

The focus of cost models is mostly in the early stages of the program. Despite the emphasis on the front end of life cycles estimates must be revisited at each significant program milestone. It is especially important to revise estimates as more accurate information about the program is made available. This provides an opportunity to revisit assumptions made by the customer and contractor as well as determining the cost impact of any changes that are requested.

Heuristic #29: Experts all disagree forever. Bound the options they are given to evaluate.

One of the important features of parametric models is that they are partially based on expert opinion. This provides an important influence to the model that is more in tune with individual experience on systems and their perceptions on the impact of project conditions on final cost. However, experts tend to frequently disagree on the relative influence of certain parameters or their definitions. There are several sampling techniques available that help them bound the options they are given to evaluate and encourage agreement like the Delphi method (Dalkey 1969).

Heuristic #30: Models are optimistic.

Unknown to many is the fact that cost models are biased towards successful outcomes and are therefore optimistic estimates. This is mostly due to the nature of historical data that was used to calibrate the model. The successful completion of projects makes them, in an indirect sense, successful cases. Furthermore, COSYSMO is strongly influenced projects that operated with high process maturity (average of Level 4) as measured by the by the Capability Maturity Model (SEI 2002).

Heuristic #31: People are generally optimistic.

As if the existence of optimistic models was not bad enough, there is ample evidence that individuals are also very optimistic particularly when it relates to estimating the length of future tasks (Buehler, et al 1994; Roy, et al 2005) and their personal abilities (Russo & Schoemaker 1992). Steps can be taken to reduce optimism bias in human judgment (Brier 1950; Koehler & Harvey 1997) but organizations should be aware of human nature and its influence on the estimation process.

Conclusion

In empirical science, if something has survived the test of time, it is a sign of quality. The collection of heuristics obtained from experience developing, using, calibrating and estimating with cost models provides such insights. Inspired by the way Zen Buddhists see opportunity in gathering knowledge from all things in life, the systems engineering community can benefit from such experiences to advance the understanding of cost models and associated challenges. Future experiences will undoubtedly help refine these heuristics as we continue the iterative process of development and refinement of heuristics for centuries to come.

Acknowledgements

This paper is a result of dialog with several colleagues over the years including Dan Ligett,

Doug Hubbard, Indrajeet Dixit, Marilee Wheaton, Tony Hart, Roger Smith, and Lacey Edwards. The work of the late Eberhardt Rechtin was a true inspiration for this paper. The lessons from my own Zen-like teacher, Barry Boehm, have also provided much enlightenment.

References

ANSI/EIA. 1999. ANSI/EIA-632-1988 *Processes for Engineering a System*.

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D. J. and Steece, B. 2000. *Software Cost Estimation With COCOMO II*, Upper Saddle River: Prentice Hall.

Box, G.E.P. 1979. *Robustness in the Strategy of Scientific Model Building*, in R.L. Launer and G.N. Wilkinson (Eds.) *Robustness in Statistics*, New York: Academic Press.

Bradbury, R. 1992. *Zen in the Art of Writing*, Bantam.

Brier, G. W. 1950. Verification of Forecasts Expressed in Terms of Probability, *Monthly Weather Review*, 75, 1-3.

Brooks, F. P. 1995. *The Mythical Man Month*, New York: Addison Wesley.

Buehler, R., Griffin, D. and Ross, M. 1994. Exploring the "planning fallacy": Why people underestimate their task completion times, *Journal of Personality and Social Psychology*, 67, 366–381.

Cockburn, A. 2000. *Writing Effective Use Cases*, New York: Addison-Wesley.

Cook, D. and Weisberg, S. (1999). *Applied Regression Including Computing and Graphics*, New York: John Wiley & Sons.

Covey, S. R. W. 1989. *The Seven Habits of Highly Effective People*, Free Press.

Dalkey, N. 1969. *The Delphi Method: An Experimental Study of Group Opinion*, RAND Corporation.

Endres, A. and D. Rombach. 2003. *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories*, London: Pearson.

Gigerenzer, G., Todd, P., & the ABC Research Group. 1999. *Simple heuristics that make us smart*. New York: Oxford University Press.

Hebron, M. 1990. *The Art and Zen of Learning Golf*, Smithtown Landing Country Club.

Herrigel, E. 1948. *Zen in der Kunst des Bogenschiessens* (Zen in the art of archery), Muenchen-Planegg: Otto Wilhelm Barth-Verlag.

Herrigel, G. L. 1999. *Zen in the Art of Flower Arrangement: The Classic Account of the Meaning and Symbolism of the Japanese Art of Ikebana*, Souvenir Press.

Hubbard, D. W. 2007. *How to Measure Anything: Finding the Intangibles in Business*, New York: Wiley.

Koehler, D. J. and Harvey, N. 1997. Confidence judgments by actors and observers, *Journal of Behavioral Decision Making*, 10, 221-242.

Koran, N. 2002. *The Zen of Tennis: A Winning Way of Life*, Koran.

Lewis, S. 1997. *Zen and the Art of Fatherhood: Lessons from a Master Dad*, Plume.

Likert, R. 1932. A Technique for the Measurement of Attitudes, *Archives of Psychology*, 140: 1-55.

O'Neill, T. and Hymel, G. 1995. *All Politics Is Local: And Other Rules of the Game*, Adams Media Corporation.

NASA (2002). Cost Estimating Handbook.

Pirsig, R. M. 1974. *Zen and the Art of Motorcycle Maintenance: An Inquiry into Values*. New York: Perennial Classics.

Rechtin, E. 1991. *Systems Architecting: Creating & Building Complex Systems*, Upper Saddle River: Prentice Hall.

Roshi, P. K. 1989. *The Three Pillars of Zen: Teaching, Practice, and Enlightenment*, Anchor.

Roy, M. M., Christenfeld, N. J. S., and McKenzie, C. R. M. 2005. Underestimating the duration of future events: Memory incorrectly used or memory bias? *Psychological Bulletin*, 131(5), 738–756.
Royce, W. 1998. *Software Project Management: A Unified Framework*, New York: Addison-Wesley.
Russo, J. E. and Schoemaker, P. J. H. 1992. Managing overconfidence. *Sloan Management Review*, 33(2), 7-17.
Sankey, J. 1998. *Zen and the Art of Stand-Up Comedy*, Theatre Arts Routledge.
CMMI. 2002. *Capability Maturity Model Integration - CMMI-SE/SW/IPPD/SS*, V1.1. Pittsburg, PA, Carnegie Mellon - Software Engineering Institute.
Shade, J. 1975. "Zen in the Art of Flying," *Air Progress Magazine*: 34-37.
Valerdi, R. 2005. *The Constructive Systems Engineering Cost Model (COSYSMO)*, PhD Dissertation, University of Southern California.
Wang, G., Valerdi, R., Ankrum, A., Millar, C. and Roedler, G. 2008. COSYSMO Reuse Extension, *18th INCOSE Symposium*.
Yamada, S. 2001. The Myth of Zen in the Art of Archery, *Japanese Journal of Religious Studies*, 28 (1–2).

Biography

Ricardo Valerdi is a Research Associate in the Lean Advancement Initiative and a Lecturer in the Engineering Systems Division at MIT. He is also the co-founder of the Systems Engineering Advancement Research Initiative (SEAri). He received his B.S./B.A. in Electrical Engineering from the University of San Diego in 1999, and his M.S. and Ph.D. degrees in Systems Architecting and Engineering from USC in 2002 and 2005. He is the author of over 50 technical publications which have appeared in several journals, including *Journal of Systems Engineering, Journal of Systems and Software, IEEE Software, International Journal of System of Systems Engineering,* and *CrossTalk - The Journal of Defense Software Engineering.* He serves on the INCOSE Board of Directors as Treasurer.